

Progressive Precision Surface Design

M.A. Duchaineau, K.I. Joy

This article was submitted to
SIGGraph 2002 29th International Conference on Computer
Graphics and Interactive Techniques, San Antonio, TX, July 21-26,
2002

January 11, 2002

U.S. Department of Energy

Lawrence
Livermore
National
Laboratory

DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

This report has been reproduced directly from the best available copy.

Available electronically at <http://www.doc.gov/bridge>

Available for a processing fee to U.S. Department of Energy
And its contractors in paper from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-mail: reports@adonis.osti.gov

Available for the sale to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Road
Springfield, VA 22161
Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-mail: orders@ntis.fedworld.gov
Online ordering: <http://www.ntis.gov/ordering.htm>

OR

Lawrence Livermore National Laboratory
Technical Information Department's Digital Library
<http://www.llnl.gov/tid/Library.html>

Progressive Precision Surface Design

Mark A. Duchaineau
Lawrence Livermore National Laboratory

Kenneth I. Joy
University of California, Davis

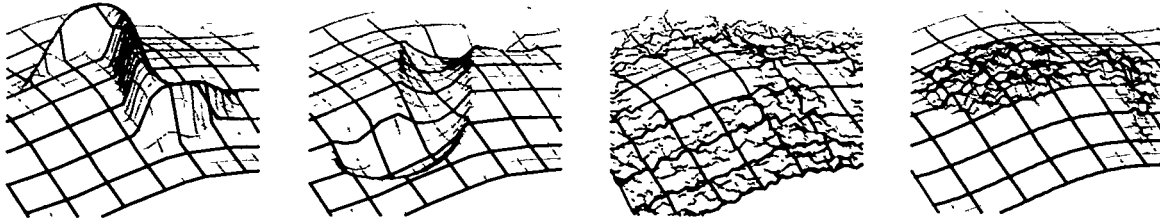


Figure 1: Precision surface design operations are made practical by our progressive wavelet decomposition. From left: pasting a template with free-form placement, precision sculpting with path and tool shape, smoothing and roughening within a roped-off area.

Abstract

We introduce a novel wavelet decomposition algorithm that makes a number of powerful new surface design operations practical. Wavelets, and hierarchical representations generally, have held promise to facilitate a variety of design tasks in a unified way by approximating results very precisely, thus avoiding a proliferation of undergirding mathematical representations. However, traditional wavelet decomposition is defined from fine to coarse resolution, thus limiting its efficiency for highly precise surface manipulation when attempting to create new non-local editing methods.

Our key contribution is the *progressive wavelet decomposition* algorithm, a general-purpose coarse-to-fine method for hierarchical fitting, based in this paper on an underlying multiresolution representation called *dyadic splines*. The algorithm requests input via a generic *interval query* mechanism, allowing a wide variety of non-local operations to be quickly implemented. The algorithm performs work proportionate to the tiny compressed output size, rather than to some arbitrarily high resolution that would otherwise be required, thus increasing performance by several orders of magnitude.

We describe several design operations that are made tractable because of the progressive decomposition. *Free-form pasting* is a generalization of the traditional control-mesh edit, but for which the shape of the change is completely general and where the shape can be placed using a free-form deformation within the surface domain. Smoothing and roughening operations are enhanced so that an arbitrary loop in the domain specifies the area of effect. Finally, the sculpting effect of moving a tool shape along a path is simulated.

CR Categories: G.1.2 [Numerical Analysis]: Approximation—Spline and piecewise polynomial approximation; Wavelets and fractals; F.2.1 [Analysis of Algorithms and Problem Complexity]: Numerical Algorithms and Problems—Computation of transforms;

E.4 [Coding and Information Theory]: Data compaction and compression; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations; Geometric algorithms, languages, and systems; Hierarchy and geometric transformations; Splines; J.6 [Computer-Aided Engineering]: Computer-aided design (CAD)

Keywords: surface editing, wavelets, progressive computation

1 Introduction

The process of designing geometric shapes via computation is a critical activity for the making of films, computer games, automobiles and many other ends. Underpinning this design activity are mathematical representations and associated algorithms that facilitate a wide variety of manipulations of shape, such as creating overall proportions, placing details, then deforming the shape or otherwise modeling various quasi-physical manipulations. Unfortunately, no single mathematical representation is known that will provide exact analytic results to all surface operations of interest. Rather than introduce more and more specialized mathematics, a recent trend has been to support many operations in a single, unified representation using approximation theory and hierarchical algorithms [DeRose et al. 1998; Lee et al. 2000; Guskov et al. 2000; Cirak et al. 2002; Perry and Frisken 2001]. However, so far only local operations have been generally supported in an efficient way for surface editing [Lounsbery 1994; Zorin et al. 1997]. In this paper, we introduce the first multiresolution framework that allows coarse-to-fine (i.e. *progressive*) computation of a broad set of non-local shape manipulations.

The key technique we introduce is the *progressive wavelet decomposition*, whereby the usual fine-to-coarse filtering and truncation is replaced by coarse-to-fine selective refinement. This switch in orientation is generally not possible unless the input data are represented and operations are evaluated in a generic hierarchical fashion, which we term *interval queries*. The abstract input interface to the progressive wavelet decomposition is therefore in the form of an interval query oracle, which the transform calls in response to selective refinement requests on the operation output. The interval query mechanism is inspired by the methods of interval analysis [Moore 1979], and the research into modeling systems built on those concepts [Snyder and Kajiya 1992; Kass 1992]. A simple, hierarchical parametric representation, *dyadic splines* [Duchaineau 1996], is used at the lowest level. A dyadic spline is defined by alternately performing B-spline refinement and adding displacement

vectors. The coarse-to-fine processing proceeds in the following phases:

1. Split a leaf of the domain-interval bintree in two, and put the (so far uncomputed) wavelet coefficients overlapping these intervals at that scale onto the active coefficient list.
2. Invoke the interval-query oracle to the target function, which provides a local Bézier patch estimate and error bound. Do this on all the domain intervals that the newly-active wavelet coefficients depend on.
3. In this neighborhood, compute the estimated values and associated error bounds of the scaling function coefficients, dyadic spline displacements, and wavelet coefficients using the appropriate local weighted-average filters.
4. Propagate improved values up to coarser resolutions if warranted, using the local wavelet decomposition filters.

The split request can be made in any order that an application chooses. A good generic ordering of these requests involves placing the domain bintree leaves on a priority queue ordered by the size of the error bounds in the neighborhood. These phases are repeated over and over until a desired accuracy is achieved or a desired time limit is reached.

We evaluate our approach with respect to six criteria:

1. **Output-sensitive computation:** Our progressive decomposition algorithm performs work proportionate to the compressed (approximated) output size. This is similar to the best algorithms in more specialized settings such as view-dependent optimization [Hoppe 1997; Duchaineau et al. 1997], multiresolution surface editing [Forsey and Bartels 1988; Zorin et al. 1997], and multiresolution painting [Berman et al. 1994], yet provides a kind of generic “plug in” architecture that eases the addition of new manipulation operations.
2. **Guaranteed error bounds:** The formulation of our transform not only is guaranteed to converge, but provides strict error bounds at every step in the progressive sequence.
3. **Fixed memory footprint:** We provide a caching system for the interval queries that allows the transform to restrict the working memory footprint to a tiny subset of the total data accessed, traversed, evaluated or output.
4. **Rate-distortion curves:** Our coarse-to-fine processing produces accuracies comparable to traditional fine-to-coarse methods at higher refinement, but suffers somewhat at coarser resolutions because the selective refinement and local approximations are based on “fuzzy” knowledge of the underlying function. In a sense this is the price that must be paid to get progressive computation, but it does not appear affect overall convergence rates.
5. **Selective refinement:** the algorithm allows applications that know where and in what order they want detail in a function domain. Interestingly, this includes feeding the output of the progressive transform into other interval-query oracles and progressive transforms, leading to a closed system for progressive computation.
6. **High-level design tools:** We devised surface design applications that are interesting in their own right but make a large point: they show the possibility of quasi-physical operations that more closely match the intuition gained from non-digital model building, as opposed to the tedium of “pulling on the

control net” by hand. In a sense this follows in the footsteps of the development of Computational Solid Geometry (CSG) [Requicha and Voelcker 1982], free-form deformations [Sederberg and Parry 1986], and hyperpatch modeling [Joy 1991].

2 Dyadic Spline Representation

This section will give a brief review of the dyadic spline representation, giving its formulation and the properties most critical the the progressive decomposition algorithm. Complete details are available in [Duchaineau 1996].

The general idea is depicted in Figure 2. An initial coarse grid of control points is alternately split and perturbed until some limit function is produced. The common uniform B-spline weighted averaging is used, and the perturbations are simple vector additions. The set of functions represented in this way is dense in L_p , meaning that all functions of interest in practical situations can be accurately converted to a dyadic spline.

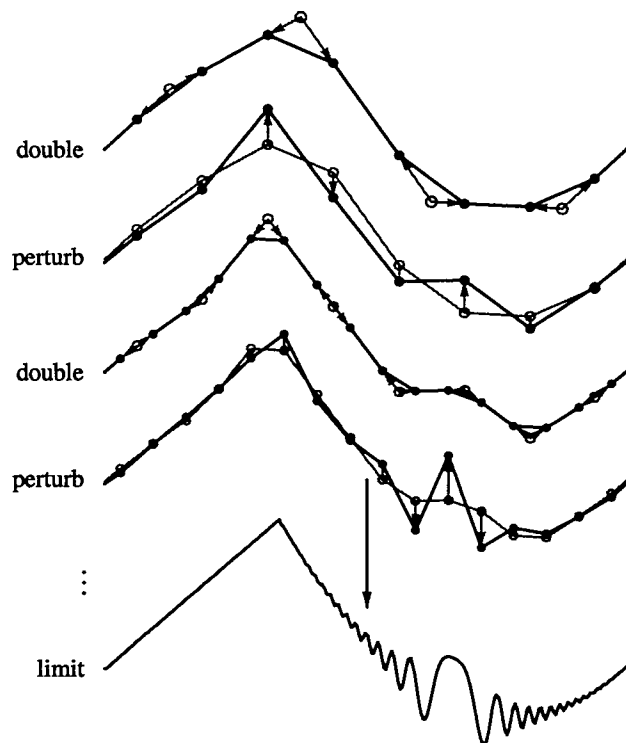


Figure 2: A dyadic spline is the limit of a sequence of doubling (B-spline refinement) and perturbing (hierarchical displacement) operations. A broad class of functions can be stored this way, but more importantly this view of a function facilitates a general form of progressive evaluation and computation.

To begin the definitions required, first recall the bintree decomposition of an m -cube. Bintrees are based on the dyadic rationals

$$\{i/2^\ell \mid i, \ell \in \mathcal{Z}\}$$

A hierarchy of one-dimensional intervals may be indexed by *level* ℓ and *position* i , as shown in Figure 3 for the subintervals of $[0, 1)$. These intervals are defined concisely as

$$I_{\ell,i} = [i/2^\ell, (i+1)/2^\ell)$$

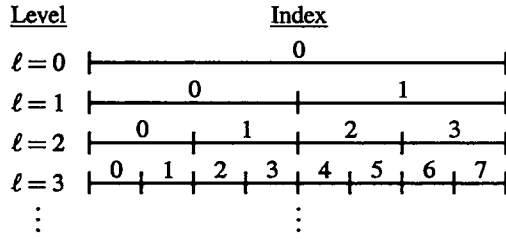


Figure 3: The dyadic interval hierarchy.

For higher dimensions, the hierarchy of one-dimensional intervals becomes a hierarchy of two- or three-dimensional intervals by splitting intervals in half along one axis at a time, as shown in Figure 4.

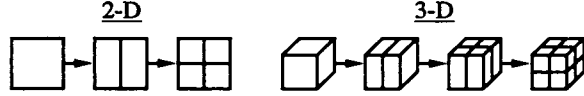


Figure 4: Bintree intervals in higher dimensions.

The intervals now have a level ℓ , current axis a , and m indices i_1, \dots, i_m :

$$I_{\ell,a,i_1,\dots,i_m} = I_{\ell+1,i_1} \times \dots \times I_{\ell+1,i_{a-1}} \times I_{\ell,i_a} \times \dots \times I_{\ell,i_m}$$

It is important to keep this hierarchy in mind, since it forms the fundamental spatial structure that all the various weighted averaging schemes make use of. In particular, *displacements* and *range positions* associated with $I_{\ell,i}$ will be denoted by $D_{\ell,i}$ and $P_{\ell,i}$ respectively.

The B-spline subdivision is given by the weighted-averaging formula

$$\begin{aligned} P_{\ell,2i} &= \sum_j \alpha_{n,j} P_{\ell-1,i+j} \\ P_{\ell,2i+1} &= \sum_j \beta_{n,j} P_{\ell-1,i+j} \end{aligned}$$

where the $\alpha_{n,j}$ and $\beta_{n,j}$ weights are given in Figure 5. A diagram of the domain intervals involved is given in Figure 6.

Degree		$j = -1$	$j = 0$	$j = 1$
$n = 1$	α	0	2/2	0
	β	0	1/2	1/2
$n = 2$	α	1/4	3/4	0
	β	0	3/4	1/4
$n = 3$	α	1/8	6/8	1/8
	β	0	4/8	4/8
$n = 4$	α	5/16	10/16	1/16
	β	1/16	10/16	5/16

Figure 5: Dyadic Refinement weights.

This is extended to include the displacements by the recurrence

$$\begin{aligned} P_{\ell,2i} &= \sum_j \alpha_{n,j} P_{\ell-1,i+j} + D_{\ell,2i} \\ P_{\ell,2i+1} &= \sum_j \beta_{n,j} P_{\ell-1,i+j} + D_{\ell,2i+1} \end{aligned}$$

Note that in this simple form (without wavelets), the dyadic spline is defined by the base control mesh P_0 and displacements D_ℓ for $\ell = 1, \dots, \infty$. Of particular importance to the coarse-to-fine progressive decomposition is the dependencies (i.e. stencils of support) in the subdivision process, shown in the cubic case in Figure 7.

The 1-D formulation just given is extended to m dimensions by the usual tensor-product mechanism, in which the one-dimensional filtering is applied along each of the axis, one at a time. The result of applying the cubic recurrence for three nonzero displacements is shown for a 2-D domain in Figure 8.

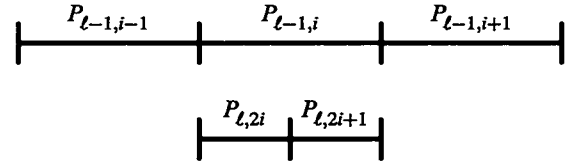


Figure 6: Dyadic Refinement intervals

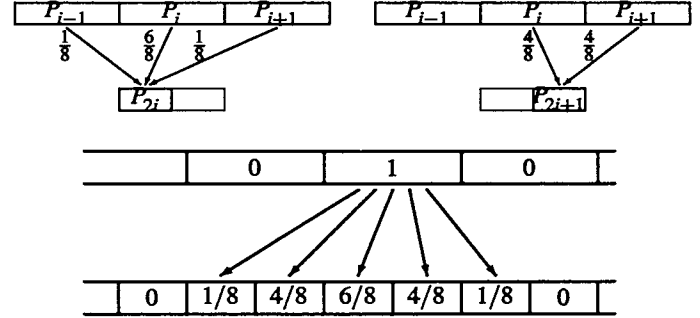


Figure 7: Dependencies in the case of cubic subdivision.

The remainder of the formulation for dyadic splines involves specifying linear operators (filters):

- S = subdivide to obtain the next finer
- F = fit points to the next finer
- C = compact the displacements
- E = expand displacements back again

The subdivision operator S has in effect already been defined, so that we can restate the dyadic spline recurrence as

$$P_\ell = SP_{\ell-1} + D_\ell$$

The subdivision filter, combined with the fit, compaction and expansion filters, form the usual wavelet decomposition bank depicted in Figure 9. Note that the C operator in effect eliminates the factor of two redundancy in the displacement representation of a function. In wavelet parlance, the $P_{\ell,i}$ values are *scaling function coefficients*, and the compacted displacements $Q_{\ell,i}$ are *wavelet coefficients*.

These three filters are defined as follows. The fit operator approximates the ideal least-squares fit operator

$$\mathbf{F}^\infty = (\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T$$

but with finite support. These operators should satisfy the following

$$\begin{bmatrix} \mathbf{F} \\ \mathbf{C} \end{bmatrix} [\mathbf{S}]\mathbf{E} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}$$

In other words, the operation of fitting and compacting the differences from prediction should be the inverse of the operation of subdividing and expanding the compacted differences. By setting the central (near diagonal) elements of \mathbf{F} to \mathbf{F}^∞ , and leaving an appropriate number as available degrees of freedom, this inverse property can be maintained by solving a tiny linear system [Duchaineau 1996]. In practice these filters have stencils of support similar to the subdivision weight masks shown earlier, and form a kind of wavelet decomposition. An alternative derivation of wavelets for computer graphics may be obtained in [Stollnitz et al. 1996].

The filters are applied in various orders depending on the operation desired. For traditional wavelet decomposition, one assumes

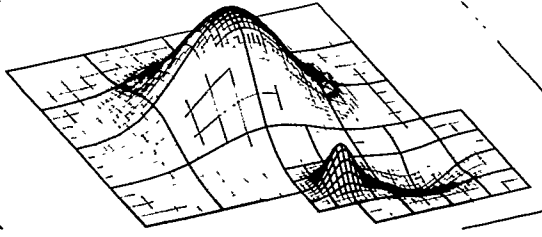


Figure 8: A 2-D dyadic spline with three nonzero displacements. The intervals that end up forming individual polynomial pieces are indicated with bold borders.

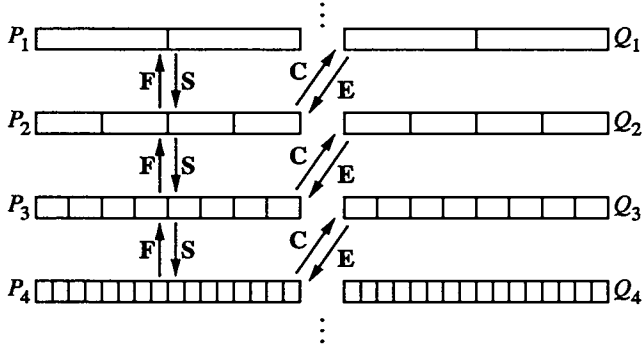


Figure 9: The dyadic spline wavelet filter bank, showing the data flow dependencies and scale relationships of the four filters F (fit), S (subdivide), C (compact) and E (expand).

some fine-level P_ℓ is given, and the computation proceeds as

$$\begin{aligned} P_3 &= FP_4 & Q_3 &= CP_4 \\ P_2 &= FP_3 & Q_2 &= CP_3 \\ &\vdots & & \end{aligned}$$

For synthesis, this is reversed:

$$\begin{aligned} P_3 &= SP_2 + EQ_3 \\ P_4 &= SP_3 + EQ_3 \\ &\vdots \end{aligned}$$

To convert from the simple dyadic spline representation (base-mesh plus displacements) to wavelets, while at the same time keeping a consistent and optimized version of the displacements, the following is used:

$$\begin{aligned} D_3 &+= FD_4 & Q_3 &= CD_4 & D_4 &= EQ_3 \\ D_2 &+= FD_3 & Q_2 &= CD_3 & D_3 &= EQ_2 \\ &\vdots & & \end{aligned}$$

This level of redundancy is useful for the formulation and implementation of surface design operations. At the end of the day, only the base mesh and wavelet coefficients are stored to disk or sent over the network.

3 Progressive Wavelet Decomposition

Looking at the wavelet filter bank in Figure 9, the computation of the wavelet coefficients $Q_{\ell,i}$ would appear to require an order of processing working from fine to coarse resolution, which we somehow need to reverse.

Let us presume that the ideal target function (the result of an editing operation, for example), is denoted $g(t)$, and that we have available an oracle that will return a local Bézier-curve estimate for $t \in I_{\ell,i}$ of $\tilde{g}_{\ell,i}(t) = \sum_j G_{\ell,i,j} B_{\ell,i,j}(t)$, where the $G_{\ell,i,j}$ are control points and $B_{\ell,i,j}(t)$ are the Bernstein basis functions of some desired polynomial degree [Farin 1999]. In addition to the local polynomial, we also need an error estimate $E_{\ell,i}$ such that $g(t) \in [\tilde{g}_{\ell,i}(t) - E_{\ell,i}, \tilde{g}_{\ell,i}(t) + E_{\ell,i}]$ for $t \in I_{\ell,i}$. We defer the discussion of how to form these estimates to the application sections, but generally this relies on having input functions in a hierarchical form, and on using variants of interval analysis [Moore 1979] to create the result of the interval query. Note that there is no need to observe continuity between $\tilde{g}_{\ell,i}$'s for our purposes; the later filtering process is immune to this and simply works best with *accurate* estimates, regardless of continuity (in effect because it is performing a convolution over these estimates with kernels that do not involve delta functions).

Suppose in a progressive decomposition that we desire to have estimates for $P_{\ell,i}$ for some intermediate level of resolution ℓ , for example at the leaves of the current bintree refinement. Given the filters F, C, E, and S we can then compute all the positions $P_{\ell',i'}$, displacements $D_{\ell',i'}$ and wavelet coefficients $Q_{\ell',i'}$ for levels $\ell' < \ell$ (values at or coarser than ℓ). So our problem is reduced to simulating what would happen if we were to perform the wavelet filtering on the infinitely resolved Bézier curves. Since the filtering process, even in this limit, is fundamentally just a linear operation of weighted averaging, we can separately precompute the infinite-limit wavelet decomposition of the Bernstein basis functions, and at runtime simply look these results up to directly compute estimated positions $\tilde{P}_{\ell,i}$ as a weighted average of the nearby estimate control points $G_{\ell',i',j}$:

$$\tilde{P}_{\ell,i} = \sum_{j,s} \beta_{s,j} G_{\ell,i+s,j}$$

for the precomputed Bernstein-basis limit fits $\beta_{s,j}$. Note that due to scale invariance these weights depend only on relative position s and basis function index j , not on the level ℓ . The estimate fit kernels $\beta_{s,j}$ are shown in Figure 10. Note that the nonzero weights are in a narrow local neighborhood.

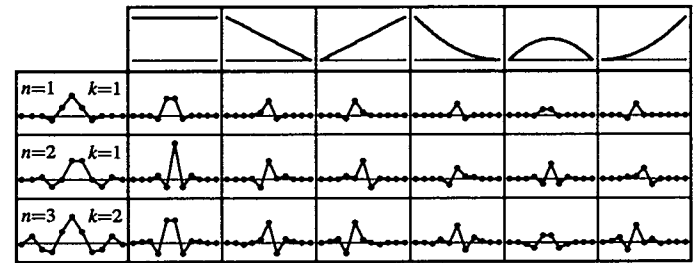


Figure 10: Estimate-fit kernels $\beta_{s,j}$ for dyadic splines of degree $n = 1, \dots, 3$ (with respective filter width parameter $k = 1, 1, 2$), and for Bernstein basis functions for degrees 0, 1, 2.

The various positions, displacements and wavelet coefficients, $\tilde{P}_{\ell,i}$, $\tilde{D}_{\ell,i}$, and $\tilde{Q}_{\ell,i}$, can now be computed using the wavelet filters. It is straightforward to obtain strict error bounds on these values since error bounds on the inputs are known and the entire process is simple linear weighted averaging [Moore 1979].

The wavelet decomposition algorithm proceeds to use this machinery to create a progressive sequence of increasingly accurate approximations to the target function $g(t)$. A pictorial example is shown in Figure 11. The target function in this case is a sequence

of “bumps within bumps” defined as the sum of transcendental functions, specifically, translated a dilated versions of the “mother bump”

$$b(t) = \begin{cases} e^{-\tan^2(\frac{\pi}{2}t)} & \text{if } t \in (-1, 1) \\ 0 & \text{otherwise} \end{cases}$$

These bump functions have closed forms for their derivatives of various degrees, and known monotonic regions, so it is straightforward to create local estimates with bounds.

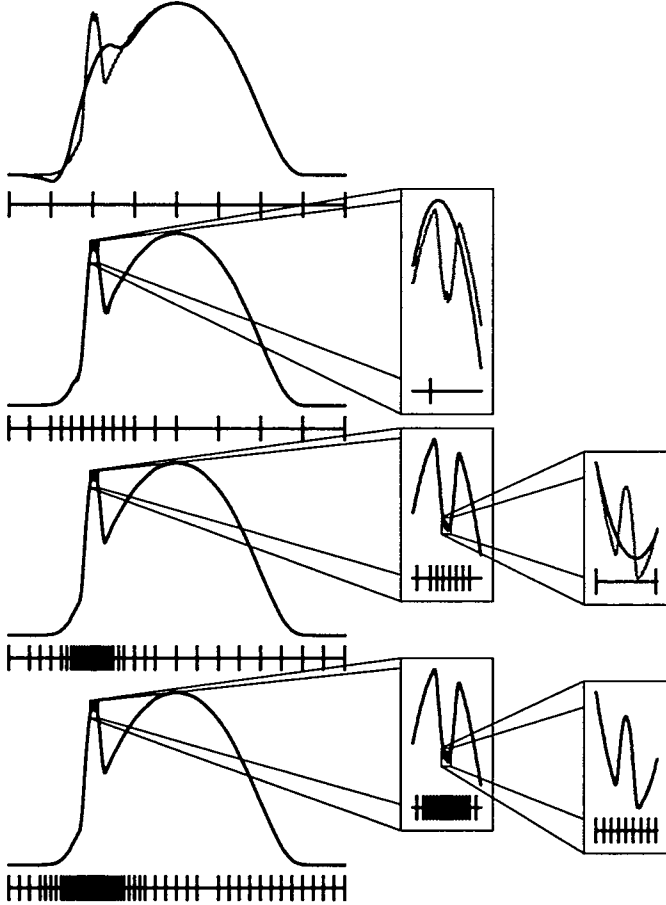


Figure 11: A target function with extremely fine-scale features (shown as a sequence of insets) is progressively computed into a wavelet decomposition under the max error norm (L_∞). The progressive transform selectively refines where errors are not guaranteed to be low, leading to a natural adaptation of the refinement around the fine features. In this example, the transform is computed over a thousand times faster than if a sufficiently fine uniform sampling of the target function were used as the starting point.

The rate-distortion curve for the example is plotted in Figure 12 (in black), compared to the usual greedy algorithm that uses fine-to-coarse processing to throw away wavelet coefficients that contribute least to the error. Note especially that the accuracies are relatively worse for the progressive transform at low numbers of coefficients (due to its fuzzy awareness of the target function), yet it “catches up” to the quality and convergence rates of the traditional greedy algorithm at higher counts.

The extension to the tensor-product setting is straightforward, as all the filtering operations just described can be performed on one axis at a time just as with subdivision. Whereas a univariate bintree decomposition $I_{\ell,i}$ was indexed by level ℓ and index i , the multivariate bintree requires an additional axis counter $a \in \{1, \dots, m\}$

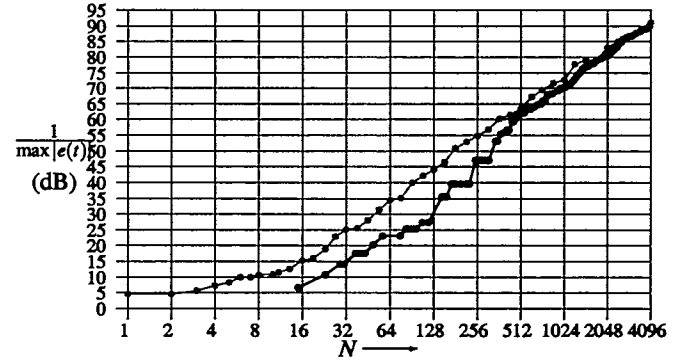


Figure 12: Coarse to fine progression of our new transform (black) is relatively low accuracy compared to the traditional greedy algorithm at low coefficient counts, yet is nearly identical at higher counts.

and multiple indices i_1, \dots, i_m . To simplify the appearance of the multivariate bintree intervals

$$I_{\ell,a,i_1,\dots,i_m} = I_{\ell+1,i_1} \times \dots \times I_{\ell+1,i_{a-1}} \times I_{\ell,i_a} \times \dots \times I_{\ell,i_m}$$

a shorthand of

$$I_{\mathcal{L},\mathbf{i}} = I_{\ell,a,i_1,\dots,i_m}$$

will be used, where $\mathcal{L} = (\ell, a)$ and $\mathbf{i} = (i_1, \dots, i_m)$. The composition $\mathcal{L} = (\ell, a)$ will be referred to as a *layer*, and is analogous to the level in the univariate case. Note that the intervals $I_{\mathcal{L},\mathbf{i}}$ still form a binary tree. The displacements are now denoted $D_{\mathcal{L},\mathbf{i}}$, and the positions $P_{\mathcal{L},\mathbf{i}}$.

An example progression for a 2-D domain with a few conical bumps is shown in Figure 13. Note how the progressive decomposition naturally adapts to the sharp features of this target function.

In the remainder of this paper we will exploit the progressive wavelet transform to create four new higher-level surface design operations.

4 Smoothing and Roughening within a General Loop

This section will describe the implementation of signal processing operations within a restricted domain area defined by a closed loop of Bézier curves. The first notion is that of performing global smoothing, which is defined for smoothing parameter ℓ_s as

$$\bar{D}_{\mathcal{L},\mathbf{i}} = \begin{cases} D_{\mathcal{L},\mathbf{i}} & \text{if } \ell < \ell_s \\ (\ell_s - (\ell - 1))D_{\mathcal{L},\mathbf{i}} & \text{if } \ell - 1 < \ell_s \leq \ell \\ 0 & \text{otherwise } (\ell_s \leq \ell - 1) \end{cases}$$

This is similar to the smoothing defined in [Finkelstein and Salesin 1994], but here extended to higher dimensions via tensor products.

For local smoothing, a generalization of the smoothing segment is needed. For this, a smoothing area is defined using the concept of a trim curve [Casale 1987], previously used in the methods for trimmed surface patches. A trim curve $c(t)$ is a continuous, periodic mapping from $t \in [0, 1]$ to the surface domain $(u, v) \in \mathbb{R}^2$. This curve encloses a domain area that will serve as the locality to be smoothed.

The smoothing operation blends between the original displacements $D_{\mathcal{L},\mathbf{i}}$ and the smoothed ones $\bar{D}_{\mathcal{L},\mathbf{i}}$. The blend factor q is defined as the fraction of $D_{\mathcal{L},\mathbf{i}}$'s interval of influence I_D that overlaps the area enclosed by $c(t)$. The computation of this overlap is

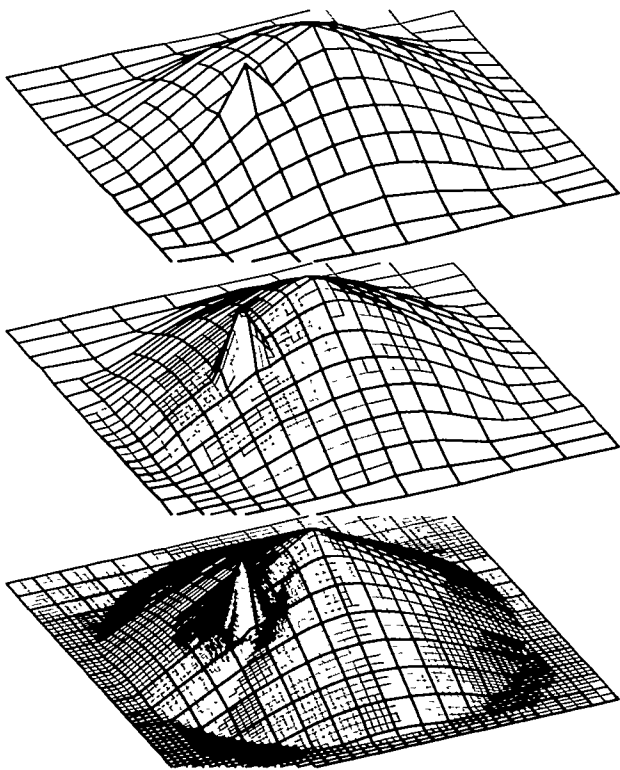


Figure 13: A demonstration of the progressive wavelet decomposition of a function as a height with 2-D domain, consisting of two overlaid conical protrusions. The wavelet coefficients are built in a coarse-to-fine sequence, shown from top to bottom.

discussed momentarily. The blend factor q is then applied as:

$$\tilde{D}_{\mathcal{L},i} = (1-q)D_{\mathcal{L},i} + q\tilde{D}_{\mathcal{L},i}$$

Some results of local smoothing are shown in Figure 14.

It is nontrivial to compute the area of overlap of an interval I and the area enclosed by a trim curve $c(t)$. However, the well-known Warnock algorithm for polygon visibility [Warnock 1969] can be adapted to this problem. Although the concern here is only for determining the area of a single “polygon” $c(t)$ within a “view window” I , the Warnock algorithm has a useful property of dividing I into smaller intervals until each interval either misses $c(t)$, $c(t)$ crosses the interval in a simple way, or the interval is small. Winding number computations are used in this algorithm to determine which intervals (or which parts of crossed intervals) are inside the trim curve. To apply the polygon techniques to a curve, the curve must be approximated by a polygon. For the purposes of interactive editing, it is sufficient to ensure that the approximation error is within a small fraction of the width of the interval I . If $c(t)$ is a dyadic spline, or is in B-spline form, standard subdivision techniques can be applied to accomplish this [Farin 1999]. In the implementation used here, “simple” crossings consist of two or fewer polygon edges, and a bintree decomposition of I is used. An example Warnock-style decomposition of a trim area is shown in Figure 15.

For roughening, we added random displacements in the manner of midpoint-displacement fractals [Miller 1986]. However, the *direction* of the displacements is taken to be in the normal direction of a smoothed version of the surface, similar to what was done in the curve case in [Finkelstein and Salesin 1994]. The offset frame for a surface displacement $D_{\mathcal{L},i}$ is defined as the two unit tangents and

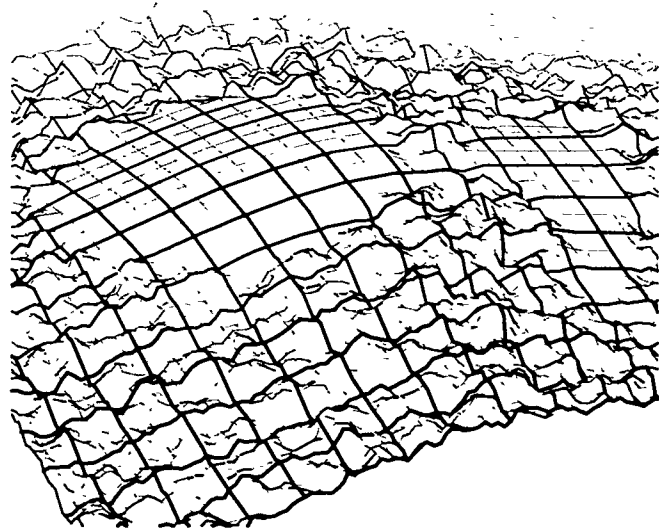


Figure 14: A formerly rough surface is smoothed within the area enclosed by a trim curve.

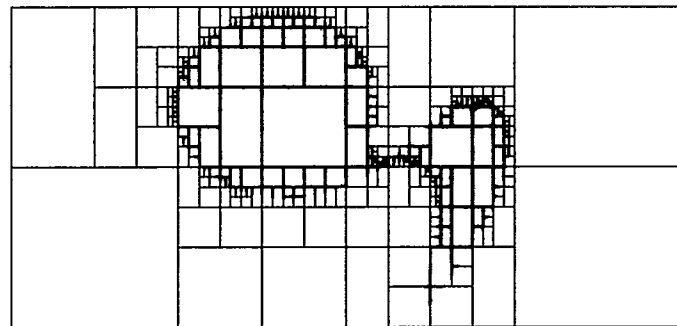


Figure 15: Decomposition of the trim-curve domain area using a variant of the Warnock algorithm.

unit normal at the point of maximum influence, taken with respect to the smoothed version of the surface.

Let $f(u,v)$ be the surface and $\tilde{f}(u,v)$ be a smoothed version of the surface for smoothing parameter ℓ_s . Then the offset coordinate frame applied to offset displacement $\tilde{D}_{\mathcal{L},i}$ is then defined as

$$A_{\mathcal{L},i} = [\mathbf{p} \ \mathbf{q} \ \mathbf{r}]$$

where

$$\mathbf{p} = \frac{\tilde{f}_u(u_m, v_m)}{\|\tilde{f}_u(u_m, v_m)\|}$$

$$\mathbf{q} = \frac{\tilde{f}_v(u_m, v_m)}{\|\tilde{f}_v(u_m, v_m)\|}$$

$$\mathbf{r} = \frac{\mathbf{p} \times \mathbf{q}}{\|\mathbf{p} \times \mathbf{q}\|}$$

and (u_m, v_m) is the domain point of maximum influence. Now the application of $A_{\mathcal{L},i}$ to $\tilde{D}_{\mathcal{L},i}$ gives the standard displacement as

$$D_{\mathcal{L},i} = A_{\mathcal{L},i} \tilde{D}_{\mathcal{L},i}$$

This gives the effect that details track the position and orientation of the smooth underlying surface.

Global roughening is produced by adding random vectors in the local smoothed-normal direction to the fine-resolution wavelet coefficients. The localization of the roughening effect is accomplished in the same manner as local surface smoothing. An example of local roughening is shown in Figure 16.

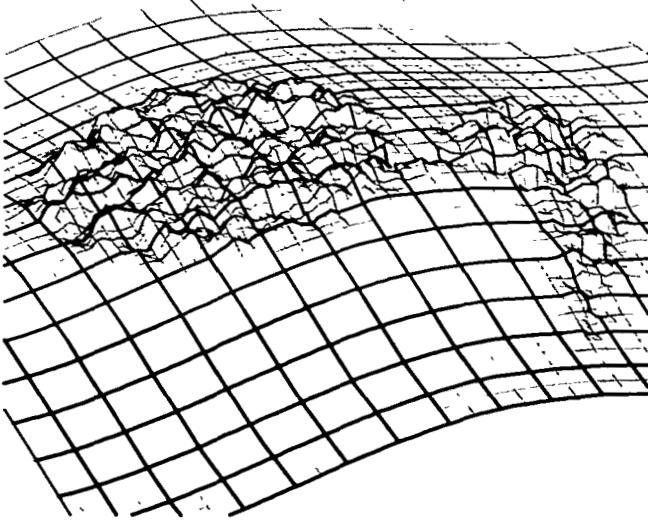


Figure 16: Random fractal offsets are made in the local smoothed normal direction. The magnitude of the offset is modulated by the Warnock in/out overlap fractions.

5 Free-form Pasting

In this section we make a more flexible version of the *Pasting* operation introduced in [Barghiel et al. 1995]. The idea is to allow an arbitrary template shape to be offset from the input surface, where the placement of the template is given by a general domain-to-domain mapping. This placement strategy is akin to a free-form deformation [Sederberg and Parry 1986] in the 2-D case. We similarly choose a bicubic Bézier patch to formulate the 2-D to 2-D mapping. The advantage of the earlier pasting formulations is that we have somewhat more general template placement and shape control, but most importantly our results are computed in progressive order to any accuracy for the precise, continuous offset definition.

Let the input surface be $f(u, v)$. Let $g(s, t)$ be a scalar-valued template function and let $h(s, t)$ be an invertible domain-positioning function into (u, v) . The basic template edit effect is defined as

$$\hat{f}(u, v) = f(u, v) + cG(u, v)$$

where c is a control vector for the *generalized basis function*

$$G(u, v) = g(h^{-1}(u, v))$$

Note that $h^{-1}(u, v)$ is only defined for $(u, v) \in h(J)$ where J is the interval domain of $h(s, t)$. When appropriate, assume that $G(u, v)$ is zero when $(u, v) \notin h(J)$. Also note that the control vector c may be derived from another control vector \hat{c} that is defined in a local offset frame A , similar to the offset-frame displacements for roughening.

The result of a template edit, $\hat{f}(u, v)$, is approximated using the progressive transform. Local estimates are formed using interval-analytic techniques. This approximation process is described shortly. An example result of template editing is depicted in Figure 17.

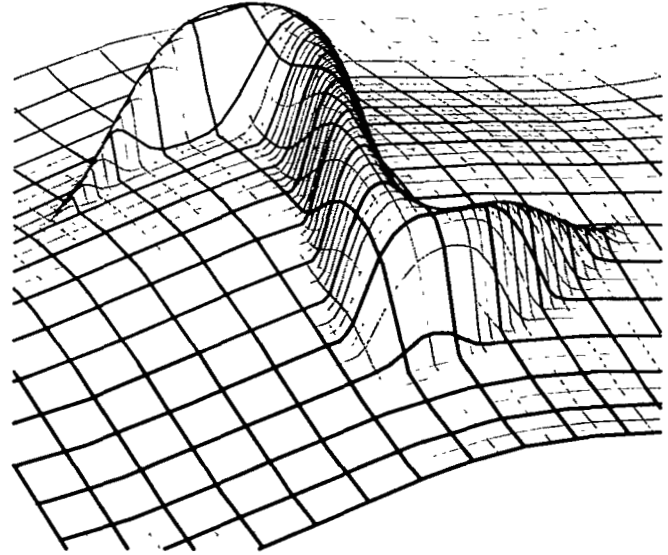


Figure 17: A simple extruded hill shape is offset from the input surface in the continuous, smoothed local normal directions. The placement of the offset template is specified with an S-shaped bicubic Bézier patch controlled in the input-surface domain.

The template edit result $\hat{f}(u, v)$ is approximated by using the progressive transform to approximate the generalized basis function $G(u, v) = g(h^{-1}(u, v))$. This approximation, denoted $\tilde{G}(u, v)$, scales a control vector c before added it to $f(u, v)$. The approximate template-edit result is $\hat{f}(u, v) = f(u, v) + c\tilde{G}(u, v)$. Applying the progressive decomposition algorithm to approximate $G(u, v)$ reduces to finding local estimates. The remainder of this section will discuss the computation of suitable local estimates.

This discussion will use first-order interval estimates throughout. To develop an estimate for $g(h^{-1}(u, v))$, an estimate will first be constructed for $h^{-1}(u, v)$ based on an estimate of $h(s, t)$. This will be composed with an estimate of $g(s, t)$ to give the desired estimate of $g(h^{-1}(u, v))$.

Let $h(s, t)$ have the first-order interval estimate

$$\tilde{h}(s, t) = H \begin{bmatrix} s \\ t \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + \delta$$

where H is an invertible 2×2 matrix, and δ is an interval in (u, v) space. Assume that this estimate holds for $h^{-1}(I)$, where I is an interval in (u, v) space. An interval estimate for $h^{-1}(u, v)$ is

$$\tilde{h}^{-1}(u, v) = H^{-1} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} s_0 \\ t_0 \end{bmatrix} + \varepsilon$$

where

$$\begin{bmatrix} s_0 \\ t_0 \end{bmatrix} = -H^{-1} \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

and where the error interval ε is chosen so that

$$\varepsilon \supset -H^{-1}\delta$$

This estimate holds for $(u, v) \in I$. The error ε may be computed as the bounding box of the image of the four corners of δ under the transform $-H^{-1}$.

Now suppose $g(s, t)$ has the estimate

$$\tilde{g}(s, t) = [g_s \ g_t] \begin{bmatrix} s \\ t \end{bmatrix} + g_0 + \gamma$$

for error interval γ , and suppose this holds for $(s, t) \in h^{-1}(I)$. Then an estimate for $g(h^{-1}(u, v))$ over I is

$$[g_s \ g_t] \left(H^{-1} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} s_0 \\ t_0 \end{bmatrix} + \varepsilon \right) + g_0 + \gamma$$

For a surface $f(u, v)$, let $\tilde{f}(u, v)$ be the smoothed version of the surface for smoothing parameter ℓ_{s1} . The tangents of this smoothed surface are normalized to give

$$\hat{p}(u, v) = \frac{\tilde{f}_u(u, v)}{\|\tilde{f}_u(u, v)\|}$$

$$\hat{q}(u, v) = \frac{\tilde{f}_v(u, v)}{\|\tilde{f}_v(u, v)\|}$$

These normalized tangents are approximated as dyadic splines (using the progressive transform) to allow the second stage of smoothing. Let $\tilde{p}(u, v)$ and $\tilde{q}(u, v)$ be the approximations to the normalized tangents, and $\bar{p}(u, v)$ and $\bar{q}(u, v)$ be the smoothed versions of these for smoothing parameter ℓ_{s2} . A final normalization and cross product gives the axis vectors of the desired offset frame

$$A(u, v) = [\bar{p}(u, v) \ \bar{q}(u, v) \ r(u, v)]$$

where

$$\bar{p}(u, v) = \frac{\tilde{p}(u, v)}{\|\tilde{p}(u, v)\|}$$

$$\bar{q}(u, v) = \frac{\tilde{q}(u, v)}{\|\tilde{q}(u, v)\|}$$

$$r(u, v) = \frac{\bar{p}(u, v) \times \bar{q}(u, v)}{\|\bar{p}(u, v) \times \bar{q}(u, v)\|}$$

The continuous offset-frame template edit becomes

$$\hat{f}(u, v) = f(u, v) + A^{-1}(u_m, v_m) A(u, v) c G(u, v)$$

where (u_m, v_m) is the domain point of maximum influence for $G(u, v)$. The transform $A^{-1}(u_m, v_m)$ is optional, but has the desirable effect that pulling the control vector c in (x, y, z) space causes the point $\hat{f}(u, v)$ to move in the same direction, as would happen when pulling the control vectors of conventional basis functions.

6 Precision Sculpting with Tool and Path

This section provides the interval-query mechanism for precisely sculpting a surface by moving a tool shape along a path in the surface domain.

A single surface "scrape" is defined by specifying tool depth in an offset-frame normal direction for each (u, v) , where depth zero occurs at a smoothed version of the surface. The offset frame tangent and normal directions $\bar{p}(u, v)$, $\bar{q}(u, v)$ and $r(u, v)$ are obtained from $A(u, v)$ as in the smoothing/roughening operations. The result of scraping is defined by the maximum of the tool depth and the depth of the original surface with respect to the smooth surface.

Let $f(u, v)$ be a given surface and $\tilde{f}(u, v)$ be the smoothed surface for some smoothing parameter ℓ_s . Let $D_T(u, v)$ be the given tool depth function, and define the surface depth as

$$D_S(u, v) = -r(u, v) \cdot (f(u, v) - \tilde{f}(u, v))$$

The result depth will be

$$D(u, v) = \max\{D_T(u, v), D_S(u, v)\}$$

Since the surface position $f(u, v)$ does not generally reside on the line through $\tilde{f}(u, v)$ in the normal direction $r(u, v)$, some means of blending from the surface to the scrape boundary is needed. A scrape boundary occurs when $D_T(u, v) = D_S(u, v)$. A simple blending method is to linearly move the surface towards the normal line as $D_S(u, v) - D_T(u, v)$ goes from positive to zero. The blend factor is defined as

$$q = \begin{cases} 0 & \text{if } D_S(u, v) - D_T(u, v) < 0 \\ \frac{D_S(u, v) - D_T(u, v)}{H} & \text{if } 0 \leq D_S(u, v) - D_T(u, v) < H \\ 1 & \text{if } H \leq D_S(u, v) - D_T(u, v) \end{cases}$$

where H is a user-supplied blend distance. The blend factor is applied to define the scrape result as

$$\hat{f}(u, v) = \tilde{f}(u, v) + D(u, v)r(u, v) + q((f(u, v) - \tilde{f}(u, v)) \cdot \bar{p}(u, v))\bar{p}(u, v) + q((f(u, v) - \tilde{f}(u, v)) \cdot \bar{q}(u, v))\bar{q}(u, v)$$

Interval estimates are used so that the progressive transform may capture the scrape result as a dyadic spline. An example of a single scrape is shown in Figure 18.

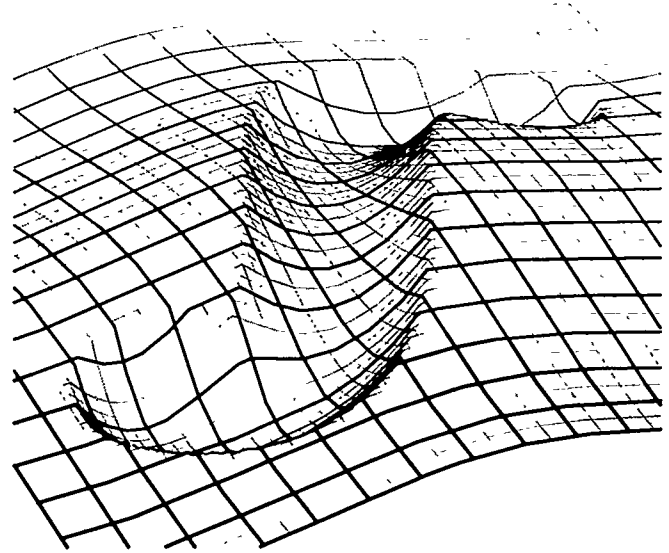


Figure 18: A single "scrape" of a tool shape along a path.

Superimposing multiple scrapes as a simultaneous operation is performed by letting the tool depth function be defined as the maximum of the individual scrape tool depth functions

$$D_T(u, v) = \max_i D_i(u, v)$$

Otherwise the formulation above remains intact. The result of two simultaneous scrapes is shown in Figure 19.

7 Conclusion and Future Work

The main discovery, in reviewing this work, is that (a) it is not obvious how to efficiently perform wavelet compression directly to the results of mathematical surface operations, yet (b) it is possible to be efficient when an intermediate interval-query oracle supplies

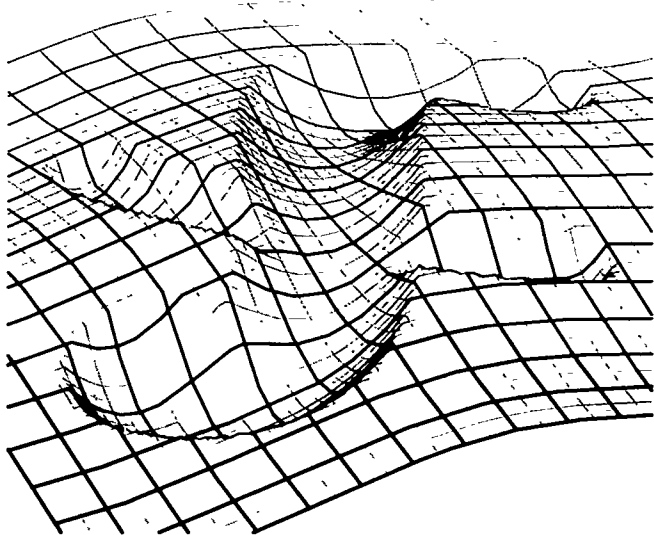


Figure 19: Two overlaid surface scrapes.

local Bézier estimates. We demonstrated by example that formulating these operations as oracle responses is tractable for a significant number of design modes that might be envisioned. We offer the following thoughts on future challenges and potential applications:

- usefulness for other wavelets

The progressive decomposition algorithm should be applicable to wavelet representations other than the dyadic splines. Only two parts of the top-down algorithm have some sensitivity to the wavelets chosen: the comparison of the wavelet approximation versus the local estimate, and the incremental, sparse updates to the wavelet coefficients as more active wavelets are added during processing. It seems likely that these issues can be solved for many wavelet schemes, including those defined on subdivision surfaces and volumes.

- tuning for various norms

The choices of which domain intervals to split and which intervals are "done" should be made with the desired norm in mind. This seems to be fairly straightforward, but has not been investigated so far.

- optimization of rate-distortion curves

A major difficulty is trying to approach the optimal rate-distortion curves, especially early in the progressive approximation process. This is hard because the local estimates only give fuzzy knowledge of the target function. Perhaps an adaptive, recursive estimation strategy could be devised that would improve this knowledge.

- general techniques for providing local estimates

In the discussions in this paper, the applications of the progressive decomposition algorithm used *ad hoc* techniques to provide local estimates to target functions. Current investigations are under way to find general, automatic methods for obtaining local estimates for a wide variety of target functions.

Acknowledgements

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

- BARGHIEL, C., BARTELS, R., AND FORSEY, D. 1995. Pasting spline surfaces. In *Mathematical Methods for Curves and Surfaces*, Vanderbilt University Press, Nashville, TN, 31–40.
- BERMAN, D. F., BARTELL, J. T., AND SALESIN, D. H. 1994. Multiresolution painting and compositing. In *Proceedings of SIGGRAPH 94*, ACM SIGGRAPH / ACM Press, Orlando, Florida, Computer Graphics Proceedings, Annual Conference Series, 85–90.
- CASALE, M. S. 1987. Free-form solid modeling with trimmed surface patches. *IEEE Computer Graphics & Applications* 7, 1 (January), 33–43.
- CIRAK, F., SCOTT, M. J., ANTONSSON, E. K., ORTIZ, M., AND SCHRÖDER, P. 2002. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *Computer-Aided Design* 34, 2 (February), 137–148.
- DEROSE, T. D., KASS, M., AND TRUONG, T. 1998. Subdivision surfaces in character animation. In *Proceedings of SIGGRAPH 98*, ACM SIGGRAPH / Addison Wesley, Orlando, Florida, Computer Graphics Proceedings, Annual Conference Series, 85–94.
- DUCHAINEAU, M. A., WOLINSKY, M., SIGETI, D. E., MILLER, M. C., ALDRICH, C., AND MINEEV-WEINSTEIN, M. B. 1997. ROAMing terrain: Real-time optimally adapting meshes. *IEEE Visualization '97* (November), 81–88.
- DUCHAINEAU, M. A. 1996. *Dyadic Splines*. PhD thesis, Dept. of Computer Science, University of California, Davis. <http://graphics.cs.ucdavis.edu/duchaine/dyadic.html>.
- FARIN, G. 1999. *NURBS: From Projective Geometry to Practical Use*. A.K. Peters, Natick MA.
- FINKELSTEIN, A., AND SALESIN, D. H. 1994. Multiresolution curves. In *Proceedings of SIGGRAPH 94*, ACM SIGGRAPH / ACM Press, Orlando, Florida, Computer Graphics Proceedings, Annual Conference Series, 261–268.
- FORSEY, D. R., AND BARTELS, R. H. 1988. Hierarchical b-spline refinement. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, vol. 22, 205–212.
- GUSKOV, I., VIDIMCE, K., SWELDENS, W., AND SCHRÖDER, P. 2000. Normal meshes. *Proceedings of SIGGRAPH 2000* (July), 95–102.
- HOPPE, H. 1997. View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH 97*, ACM SIGGRAPH / Addison Wesley, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 189–198.
- JOY, K. 1991. Utilizing parametric hyperpatch methods for modeling and display of free-form solids. In *SMA '91: Proceedings of the First Symposium on Solid Modeling Foundations and CAD/CAM Applications*, ACM Press / ACM, held June 5–7, 1991 in Austin, Texas, USA., 245–254.
- KASS, M. 1992. Condor: Constraint-based dataflow. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, vol. 26, 321–330.
- LEE, A., MORETON, H., AND HOPPE, H. 2000. Displaced subdivision surfaces. In *Proceedings of SIGGRAPH 2000*, ACM Press / ACM SIGGRAPH / Addison Wesley Longman, Computer Graphics Proceedings, Annual Conference Series, 85–94.

- LOUNSBERRY, M. 1994. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, Dept. of Computer Science and Engineering, U. of Washington.
- MILLER, G. S. P. 1986. The definition and rendering of terrain maps. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, vol. 20, 39–48.
- MOORE, R. E. 1979. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia.
- PERRY, R. N., AND FRISKEN, S. F. 2001. Kizamu: A system for sculpting digital characters. In *Proceedings of SIGGRAPH 2001*, ACM Press / ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 47–56.
- REQUICHA, A. A. G., AND VOELCKER, H. B. 1982. Solid modeling: a historical summary and contemporary assessment. *IEEE Computer Graphics & Applications* 2 (March), 9–22.
- SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, vol. 20, 151–160.
- SNYDER, J. M., AND KAJIYA, J. T. 1992. Generative modeling: A symbolic system for geometric modeling. In *Computer Graphics (Proceedings of SIGGRAPH 92)*, vol. 26, 369–378.
- STOLLNITZ, E. J., DEROSE, T. D., AND SALESIN, D. H. 1996. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA.
- WARNOCK, J. E. 1969. A hidden-surface algorithm for computer generated half-tone pictures. Tech. Rep. TR 4–15, NTIS AS-733 671, Computer Science Department, University of Utah.
- ZORIN, D., SCHRÖDER, P., AND SWELDENS, W. 1997. Interactive multiresolution mesh editing. In *Proceedings of SIGGRAPH 97*, ACM SIGGRAPH / Addison Wesley, Los Angeles, California, Computer Graphics Proceedings, Annual Conference Series, 259–268.